**1: Chapter 7. Know Stevens setjmp and longjmp**

Setjump acts as an anchor for jumping. This means that you use setjmp to set the anchor. Then, later on, if there is an error and you want to return to the anchored point, you use longjmp. Longjmp is passed the same environmental variable as setjmp to identify that that is the setjmp that longjmp links to, and an integer which is what setjmp actually returns. This allows multiple longjmps to link to one setjmp, but each with different terminating conditions.

**2: What are the typical uses of the following Unix directories. Give an example of a file in each directory**

/usr/bin-executable files go here, something like sublime

/usr/var-user constantly changing files like emails

/bin-contains binaries. And fundamental utilities like ls

/dev/null-it is the black hole of files. Write to it, nothing is actually written, read=EOF

/home-home directories of users

/dev-peripheral devices go here flash drives currently plugged in, mp3 players

/mnt-here one can find other partitions, CDROM drives, Floppies. Hardware that’s mountable

/var-this is home to constantly changing files like process-ID lock files

**3: What is a search path? Where is that information kept when a process is created via fork and execvp()? Can a process modify its search path before the execvp? Explain.**

The search path helps in the location of programs. It’s a list of directories that may contain programs the user wants to run. The directories are examined when the user types the name of a program. When the executable is found, it’s executed. A process cannot modify its search path, but in CPP one can modify the search path.

**4: Give an example of a problem that requires concurrency and can be solved with either processes or threads. Give an example of a problem whose solution requires processes.**

A problem specifically requiring processes is the obvious shell problem.

Databases are an example where concurrency is needed. Lost updates, bad reads, and incorrect summaries are a problem without concurrency.

5: Describe the substructures of a disk partition that has been used to create a Unix file system. What information is in a v-node or i-node? Explain what a ‘hole’ in a file is, and why the commands %ls and %du give such different results.

The i-node is the actual representation of a file. Directories contain a list of file-names and i-nodes.

The v-node contains information about the file pointers. It allows use of different file systems.

**6: There are 6 Unix System Primitives including pipes and Signals. Define each and give an example of appropriate use.**

Signals: signal’s allow processes to communicate states to each other. If the user hits the interrupt key, or a process terminates prematurely, it sends signals to let other processes associated to it know.

Pipes:

**7: The following acronyms arise in the discussion of Unix/C standards and history. Create a paragraph that coherently uses these acronyms in context.**

There are many operating Systems which are derived from Unix. There are those which comply with the SUS, also know as POSIX, a set of standards specified by the IEEE’s 1003.\* specification set to maintain compatibility among operating systems. Since then, many standards have developed including FIPS, IEC, COSE, ANSI, and SPEC1170 in response to IEEE charging fees for POSIX use. There are a few operating systems which comply completely with SUS like BSD. Then, there are those which don’t fully, and are known as UNIX-like. These are the common Linux and its descendents, FreeBSD and BSD’s descendents, Solaris, MINIX, and others. Linux, being the most notable, is commonly misconceived as GNU/Linux. GUN was a separate project to develop a free and open operating system. GNU is not complete, and used the Linux kernel to aid development speed. A common item every Linux distribution uses by default it the GCC compiler system. This was developed by the GNU project and conforms to ISO C, the ISO’s C compiler standards. GCC has since been expanded to compile C++, and many front-end modifications were made for many other languages.

ISO C-a set of standards C compilers adhere to defined by the International Standardization Org

ISO-International Standardization Organization

IEC-ISO for electronics

IEEE-ISO for engineering

BSD-An OS which comes from Unix compliant with SUS

FreeBSD-a descendent of BSD which is Unix-Like

FIPS-american ISO for government contractors

POSIX-Portable Operating System Interface standards specified by IEEE to maintain compatibility among OSs.

POSIX+-I don’t think this exists. I would think a more strict version of POSIX, though.

POSIX.1-POSIX standards for Core services.

ANSI-Similar to IEEE, but American

GCC-GNU Compiler Collection. A compiler system originally for C, then C/C++, with front-ends for many other languages.

GNU-Richard Stallman’s OS intended to replace UNIX with a microkernel but didn’t work

IEEE 1003.\*-The standards referred to a POSIX

SUS-Single Unix Specification. A collective of standards for OSs qualifying as UNIX OSs.

SUSv3-major revision of SUS, POSIX:2008 is now the newer one

XSI-core application programming interface conforming to SUS

COSE-started to create open, unified OS standards similar to SPEC1170

SPEC 1170-created by vendors to be a free POSIX because IEEE charges for POSIX

Linux-a Unix-Like operating system

MINIX-a Unix-Like operating system

Solaris-a Unix OS made by SUN, turned into OpenSolaris

**8: Write pseudo-code for the shell and explain the proper order in which dup2, exec, pipe, and fork are used.**

if cd, then change directory

else, process id = fork.

If the number of pipes is 0

If we redirect output, open the file, then dup2 the standard out to it

If we redirect input, open file, then dup2 the standard in from it.

Then exec the argument

Else we have more pipes

If this is the parent, pipe the left.

Check for ouput redirection, open that file, then dup2 the standard out to it

Close the left pipe.

Dup2 standard in to the left pipe

Execvp the argument

If this is the child,

Right pipe becomes left pipe, then dup2 the standard out to the right pipe.

Close the right pipe.

Fork

If we’re now in the parent

close the left pipe, then dup2 the standard in to it

execvp the argument

if we’re now in the child

right pipe becomes left pipe

if there’s input redirection, dup2 the standard in to the filepointer

exec the argument

The proper order is fork, pipe, dup2, exec. This is because we must first fork, then set up the pipes with dup2 being used, and then we can exec the argument inside and it will be piped accordingly.

**9: Explain makeargv.c and why a char \*\*\* is needed**

Makeargv takes a string input, and chops it up by spaces to make the processing of the input easier.

Char\* contains the location of a character array, or string. Char\*\* contains the location of arrays or strings. The reason Char\*\*\* is used because if contains multiple arrays of arrays of strings. This is because it tracks the actual location in the original input string, and the contents of that location. Alternatively, I believe this could be done with a char\*\* and an int, but that would be more difficult to implement.

10: Know these programs.

Directory Traversal

Creating/ Using a Random Access File

Dynamically creating arrays of structs

Structs, Pointers to structs, how to access fields using each

**11: Typical String library routines**

String libraries contain within them ways of handling Strings, but do so with chars and char\* and char\*\*, because Strings are a made-up variable in the sense that it is built using other variables and manipulating those similar to the Newton in physics.

**12: Examples of thread creation, synch, and mutex()**

Synchronizing threads is key so that neither thread destroys the other’s actions. A mutex works similarly by locking access to a shared resource and unlocking after use.

**13: RingBuffer, threaded and unthreaded.**

Threaded: before reading, writing, use mutex to lock the process. Then after, unlock the process.

Unthreaded is just the normal ringbuffer.

**14: Explain strtok()’s thread unsafeness. Can two processes use strtok() safely, or or two threads within a process not use it safely. Explain using the Process Memory Model(Chapter 7).**

It is thread unsafe within a program. Two processes can use strtok independently as they have 2 versions of the data in question. Within one process, though, strtok is thread unsafe because it advances that process’s pointer to the next item in the heap of data, which could cause the other thread to miss out on data.

**15: Is readdir()thread unsafe?**

Yes, it is. Readdir() reads the next directory, so if two threads are reading directories sequentially, one could readdir, and the other thread would skip that directory and go to the next one, because that one has been read already

**16: What is reentrancy?**

Interrupted during use, and then safely veing called again. This is common with signals and threads

**17: What state does an exec’d process maintain?**

The same as the current program. Execing merely replaces the current process with a brand new program from heap.

**18: What state does a forked process maintain?**

It is an exact copy of the current running program, and continues to operate as that program, only with the constraints of it being a child process.

19: Know the IPC primitives and their use.